Secure and Efficient Client-Side Data Deduplication with Public Auditing in Cloud Storage

Qianlong Dang, Hua Ma, Zhenhua Liu, and Ying Xie
(Corresponding author: Qianlong Dang)

School of Mathematics and Statistics, Xidian University
No.2, South Taibai Road, Xi'an Shaanxi 710071, P.R. China
(Email: xidianqldang@163.com)

(Received Nov. 06, 2018; Revised and Accepted Feb. 7, 2019; First Online June 14, 2019)

Abstract

In this paper, we propose a secure and efficient client-side data deduplication scheme with public auditing. In the process of deduplication, the proposed scheme improves the probability that the cloud server detects the missing blocks by eliminating the aggregated proofs structure. Meanwhile, we combine the oblivious pseudo-random function protocol with proxy re-encryption technology to implement key distribution without online data owners or the authorized party. Moreover, during public auditing, proxy re-signature technology is utilized to require only one auditing tag for each data block. For data deduplication, the proposed scheme is a zero-knowledge proof of knowledge assuming that the Discrete Logarithm (DL) problem is hard. In addition, the symmetric key can not be recovered by the cloud server or malicious users. And security analysis indicates that our scheme is secure against adaptive chosen-message attack under the Computational Diffie-Hellman (CDH) assumption during public auditing. Finally, the performance evaluation demonstrates that the proposed scheme is practical and efficient.

Keywords: Cloud Storage; Key Distribution; Proof of Ownership; Public Auditing; Secure Deduplication

1 Introduction

In cloud storage services, clients outsource data to a remote storage and access the data whenever they need the data. Recently, owing to its convenience, cloud storage services have become widespread, and it can provide resource-constrained users with convenient storage and computing services [5, 16, 21]. Although the cloud storage offers many advantage, it also brings a huge storage burden and some security challenges such as data integrity [23].

Since cloud storage service is increasingly used, a large amount of data is gathered into the cloud server. IDC predicts that the cloud data will reach 44ZB in 2020. A recent survey conducted by Microsoft [22] indicates that about 90% of data stored in the cloud are duplicated copies. Regarding storage efficiency, commercial cloud storage services, such as Dropbox, Wuala and Bitcasa, adopt deduplication technique to store one copy of each data and refer other duplicates to this stored copy. However, several security threats potentially exist during deduplication [1, 6, 14, 15, 24]. For instance, if a malicious user needs to gain access to the data that already exists in the cloud server, he can pass the verification by only owing the hash value of the data rather than the original data. It is obvious that the cloud server cannot distinguish whether user indeed possess the data only through matching its hash value. Therefore, how to convince the cloud server that the user indeed possesses original data becomes an important problem.

As a promising approach, message-locked encryption (MLE) [3] was used as client-side deduplication schemes [14, 15]. However, almost all of these schemes adopt deterministic encryption method and are vulnerable to brute force dictionary attacks. To solve this issue, some schemes encrypt the data with a randomly selected symmetric key, and the first uploader distribute the symmetric key to subsequent uploaders by adopting the proxy re-encryption technology. Unfortunately, all existing key distribution processes require the assistance of online data owners or the authorized party. In order to improve security and efficiency, it is essential to consider how can the deduplication scheme implement key distribution without the assistance of online data owners or the authorized party.

When clients use cloud storage services, they have hopes of guaranteeing the completeness of cloud storage data [4, 11, 13, 18, 30]. Accordingly, we need an efficient way to check the integrity of data in remote storage. Clients decide to authorize the task of auditing to a third party auditor (TPA), which enables that clients can efficiently perform integrity verifications even without the local copy

of data. However, these integrity auditing schemes rarely consider secure client-side data deduplication. Therefore, it is essential to combine secure client-side deduplication with integrity auditing.

In this paper, aiming at solving both storage efficiency and data integrity, we concentrate on how to design a secure and efficient client-side data deduplication scheme with public auditing. Inspired by a proof of ownership protocol [28] and a key distribution process [6, 28], we will propose an efficient client-side data deduplication and public auditing scheme which achieves a better trade-off between functionality and efficiency through improving Liu *et al.*'s auditing scheme [20]. Our main contributions can be summarized as follows.

- We utilize the aggregated proofs structure and zero-knowledge proof for proof of ownership, which improves the probability that the cloud server detects the missing blocks. Meanwhile, we prove that the proof of ownership scheme is sound, complete and zero-knowledge.
- The proposed scheme integrates the oblivious pseudo-random function (O-PRF) protocol with proxy re-encryption technology to implement key distribution. In the process of data deduplication, the proposed scheme does not require the assistance of online data owners or the authorized party. The process shows that the symmetric key of data can not be recovered by the cloud server or malicious users.
- By adopting proxy re-signature technology, subsequent uploaders can verify integrity of the cloud storage data in the proposed scheme. We prove that the proposed auditing scheme can guarantee the correctness and unforgeability. Finally, the performance evaluation demonstrates that the proposed scheme is practical and efficient.

The rest of this paper is organized as follows. A review about some related works is given in Section 2. Some preliminaries are presented in Section 3. Section 4 defines system and security model. The concrete construction of secure and efficient client-side deduplication scheme with public auditing is detailed in Section 5. Section 6 analyzes the security of our scheme. Section 7 presents the performance evaluation. Finally, we conclude the paper in Section 8.

2 Related Works

With the development of cloud computing, a rising number of enterprises and organizations choose to outsource their data to the cloud server. Then, the large amount of data is gathered in the cloud server, which will bring huge storage overhead to the cloud server. Therefore, client-side deduplication technology is introduced to solve data redundancy problems. During a client-side deduplication system, after receiving the hash value of data from the user, the cloud server checks whether the duplicate exists in cloud storage. Nonetheless, Halevi *et al.* [10] explained several security attacks that may occur in client-side deduplication systems. Moreover, Halevi *et al.* [9] proposed the concept of proof of ownership (PoW). The purpose of proof of ownership is to better verify that a client owns the entire data instead of owning partial data. Some scholars have proposed a variety of PoW schemes [8, 25–27]. However, when data ownership is verified, the existing schemes are based on the hash of the data rather than the original data. That is to say, the clients could be accepted by the cloud server as data owners with the hash of the data, even if they do not have original data. To address this issue, Yang *et al.* [28] verified the data ownership by the original data block. Nevertheless, this scheme has a low probability of detecting missing blocks.

The vast majority of client-side deduplication schemes [14, 15] adopted message-locked encryption technology to achieve data deduplication. However, these schemes are vulnerable to brute force dictionary attacks. Yang *et al.* [28] presented a scheme that the first uploader encrypts the data by a random symmetric key and utilizes the proxy re-encryption technology to distribute symmetric keys to subsequent uploaders. This scheme requires the online data owner to assist key distribution. In addition, Ding *et al.* [6] introduced an authorized party to complete key distribution in the client-side deduplication scheme.

When a client stores data in the cloud server, the user loses the right of managing the data. Consequently, it is very vital for users to check the integrity of cloud storage data in time. Recently, some scholars have proposed a variety of auditing schemes [7, 12, 29]. However, these schemes fail to achieve secure deduplication. Li *et al.* [17] proposed an integrity auditing scheme for encrypted deduplication storage, which introduced a third-party cluster to generate the same signature tag for duplicate data, but brought some data privacy issues. In order to resolve this problem, Liu *et al.* [20] used the MLE and proxy re-signature to actualize the deduplication of auditing tags among users, which can protect data privacy and generate one tag for the identical data block. However, due to the adoption of message-locked encryption technology, this scheme is vulnerable to brute force dictionary attacks. In addition, this scheme adopts server-side deduplication, which causes huge network bandwidth consumption.

3 Preliminaries

We now explain some preliminary notions that will form the foundations of our scheme.

3.1 Bilinear Pairings

Let G_1 and G_T be two multiplicative cyclic groups of the same prime order q. Let $e : G_1 \times G_1 \to G_T$ denote a bilinear map [28] constructed with the following properties:

1) Bilinearity: For all $a, b \in Z_q^*$ and $g_1, g_2 \in G_1$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2) Non-degeneracy: There exists a point g_1 such that $e(g_1, g_1) \neq 1$.
3) Computability: $e(g_1, g_2)$ for any $g_1, g_2 \in G_1$ can be computed efficiently.

3.2 Complexity Assumptions

Definition 1. *(Discrete Logarithm (DL) problem [6]) Given $g \in G_1$ and $y = g^x$, where x are selected uniformly at random from Z_q^*, it is hard to get x.*

Definition 2. *(Computational Diffie-Hellman (CDH) problem [6]) Given a group G_1 with generator g and elements $g^x, g^y \in G_1$, where x, y are selected uniformly at random from Z_q^*, it is hard to compute the value of g^{xy}.*

3.3 Oblivious Pseudo-Random Function Protocol

Oblivious pseudo-random function protocol (O-PRF protocol) is introduced in DupLESS [2]. This protocol generates secret value by interacting between the key server and the user instead of deriving the secret value from the data directly.

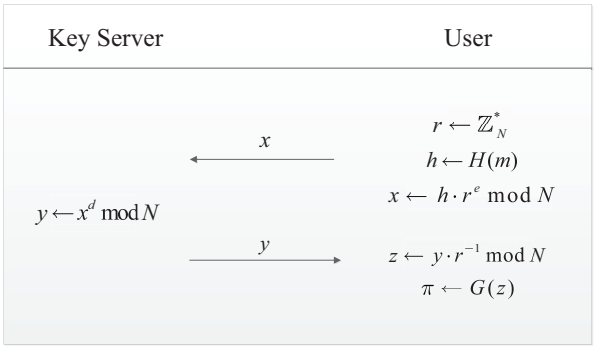

Figure 1: O-PRF protocol

Figure 1 illustrates the O-PRF protocol based on RSA blind signatures. The key server has secret key d and public key e where $ed \equiv 1 \bmod \Phi(N)$. $H : \{0,1\}^* \rightarrow Z_N^*$ and $G : Z_N^* \rightarrow \{0,1\}^*$ are two secure hash functions. The interaction process is as follows. (1) The uploader calculates the hash value $h = H(m)$ of the data and selects a random value $r \in Z_N^*$. Moreover, the uploader computes the blinded hash $x = h \cdot r^e \bmod N$ and sends x to the key server. (2) Upon receiving x, the key server computes $y = x^d \bmod N$ and sends y to the uploader. (3) The uploader calculates $y \cdot r^{-1} \bmod N$ and obtains secret value z. Finally, the uploader computes the secret value $\pi \leftarrow G(z)$.

The cloud server does not have the hash value h of data, so it cannot generate the secret value π. Moreover, the secret value generation process will not disclose any information. A malicious user who does not have a secret key d, therefore, cannot generate secret value π. It allows encryption to be secure against the brute force attacks even for predictable message set.

3.4 Aggregated Proofs Structure

During the verification process, the aggregated proofs [19] can improve verification efficiency and save network bandwidth. By using the idea of aggregated proofs, we design the aggregated proofs structure (As shown in Figure 2). The first uploader uploads the original proofs to the cloud server. Firstly, the original proofs are multiplied by the selected coefficient to obtain the first-level proofs. Secondly, the second-level proofs are generated by multiplying two adjacent terms in the first-level proofs. Thirdly, the third-level proofs are generated by multiplying two adjacent terms in the second-level proofs. Finally, the j-level proofs are calculated.

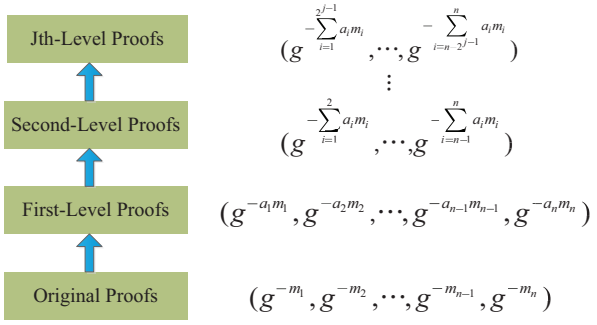

Figure 2: Aggregated proofs structure

In the PoW protocol, many scholars adopt the Merkel hash tree to verify ownership of the data. These schemes do not verify the data ownership based on accessing of the original data. In other words, the verification is based on the Merkle hash tree which is built over one hash of the original data rather than the original data. Therefore, a malicious user could pass the PoW verification of client-side deduplication if he could get the hash value of the data. However, the aggregated proofs structure verifies the data ownership by the original data block. Moreover, the proposed scheme improves the detection rate and saves network bandwidth by using the aggregated proofs structure.

4 Problem Statement

4.1 System Model

The system model of the proposed scheme is described as Figure 3, which includes five entities: cloud service provider, key server, third party auditor, first uploader and subsequent uploaders.

- **Cloud Service Provider (CSP)**: The CSP stores the encrypted data uploaded by the first uploader and performs deduplication operations with subsequent

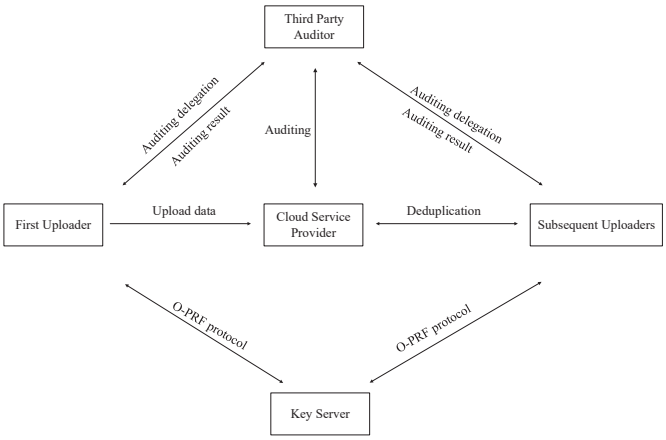

Figure 3: System model of our scheme

uploaders. In addition, when uploaders delegate a third party auditor to audit cloud storage data, the CSP generates corresponding auditing proofs.

- **Key Server (KS)**: In the O-PRF protocol, upon receiving the blinded hash value from the uploader, the KS signs it using the oblivious pseudo-random function. As a result of the interacting, the ciphertext is secure against the brute-force attack by known set.
- **Third Party Auditor (TPA)**: When the uploader sends auditing delegation, the TPA invokes the integrity auditing protocol by sending the CSP a challenge. On receipt of the proof from the CSP, TPA verifies the target data integrity and notifies the client of the result.
- **First Uploader**: The first uploader interacts with the KS to generate a secret value of the data by the O-PRF protocol. Then, the first uploader generates ciphertext and uploads it to the CSP. When the first uploader wants to check the integrity of the cloud storage data, he sends the auditing delegation to the TPA.
- **Subsequent Uploaders**: The subsequent uploaders interact with the KS to generate a secret value of the data by the O-PRF protocol. Moreover, the subsequent uploaders perform deduplication operations with CSP and send the auditing delegation to TPA.

4.2 Threat and Security Model

We give the threat model of the proof of ownership process and the key distribution process. Moreover, the security model of integrity auditing scheme is defined.

4.2.1 Threat Model

The existing deduplication schemes [6,28] did not construct a formal security model, but only gave some threat models. Therefore, the threat model for a malicious user and the cloud servers is constructed as follows.

In PoW protocol, a malicious user is to pass the PoW challenge for a message m while he only knows some partial information of m. Suppose that the malicious user possesses several data blocks and the hash value of m . Therefore, the malicious user can attempt to forge proofs for passing the PoW challenge. On the other hand, the cloud server wants to get some information about the data during the ownership verification process.

In key distribution process, a malicious user owns the secret value of the data, but he is an unauthorized user. The malicious user can attempt to get the symmetric key of the data. Moreover, since the cloud server re-encrypts the ciphertext of the data, the cloud server also attempts to obtain the symmetric key of the data.

4.2.2 Security Model

As for the security of integrity auditing, similar to the existing definition [20], we consider the probability that the cloud server can convince the user that the cloud storage data is stored correctly while the cloud storage data has been corrupted or deleted. We say that the proposed scheme is secure against an adaptive chosen-message attack. The specific process of this game is as follows.

Setup: We divide users into normal users and malicious users. For the l normal users and l' malicious users in the system, the challenger performs KeyGen algorithm to generate user-associated public/private key pairs $(pk_{nu}, sk_{nu})_{nu\in[1,l]}$ and $(pk_{mu}, sk_{mu})_{mu\in[1,l']}$. Finally, the normal users' public keys and the malicious users' public/ private keys are sent to the adversary $\mathcal{A}$.

Query 1: The adversary $\mathcal{A}$ can adaptively query **SecValGen** to obtain message-related public/private key pairs $(pk_{\pi'_\omega}, \pi'_\omega)_{\omega\in[1,o']}$ for o' data. Then, $\mathcal{A}$ queries **Rekey** and gets re-signature keys $rk'_{nu,m}$ and $rk'_{mu,m}$. Finally, $\mathcal{A}$ adaptively queries **TagBlock** as follows.

The adversary $\mathcal{A}$ chooses a block E'_1 and sends it to the challenger for the tag under message-related public key $pk_{\pi'_\omega}$. The challenger calls **TagBlock** algorithm and sends $T'_{1,\omega}$ back to $\mathcal{A}$. $\mathcal{A}$ continually queries the tags on blocks $E'_2, \ldots, E'_{n'}$ under $pk_{\pi'_\omega}$, and the challenger responds $T'_{2,\omega}, \ldots, T'_{n',\omega}$ accordingly. In the end, $\mathcal{A}$ stores the blocks and their tags.

Query 2: The adversary $\mathcal{A}$ can adaptively query **SecValGen** to obtain message-related public key $(pk_{\pi'_\omega})_{\omega\in[1,o]}$ for o data. $\mathcal{A}$ then queries **Rekey** and gets re-signature keys $rk'_{nu,m}$ for $(1 \le nu \le l, 1 \le \omega \le o)$. Finally, $\mathcal{A}$ adaptively queries **TagBlock** on blocks $E_1, E_2, \ldots, E_n$ as the case in query 1.

Challenge: The challenger requests $\mathcal{A}$ to provide a proof of possession for $\{E_i\}_{i\in I\subseteq[1,n]}$ determined by a challenge $Chal$ under the user public key pk_{nu}.

Forge: The adversary $\mathcal{A}$ outputs a possession proof P.

If CheckProof returns 1, then the adversary $\mathcal{A}$ wins this game.

Definition 3. *We say that a data integrity auditing scheme is secure, if for any probabilistic polynomial time adversary $\mathcal{A}$ who does not possess all of the challenged data blocks, the probability that $\mathcal{A}$ succeeds in the above game is negligible.*

4.3 Working Graphs

As shown in Figure 4, we describe in detail the whole process of our scheme.

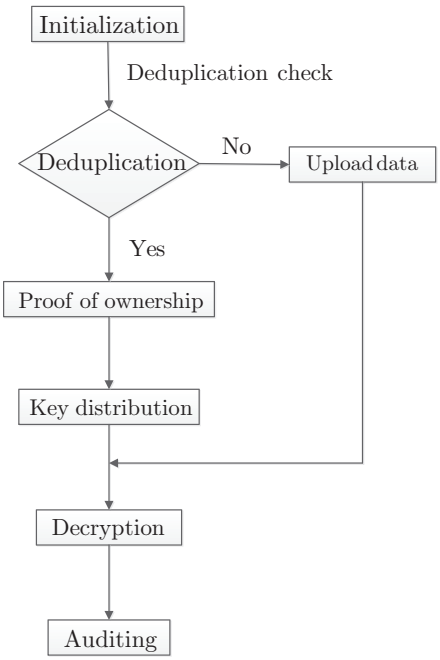

Figure 4: The working graphs of our scheme

In the initialization phase, the system generates the public parameters. The cloud server and the uploader generate their key pairs. In addition, uploaders interact with the key server to generate a secret value of the data. When a uploader uploads a tag to the cloud server, the cloud server performs deduplication detection on the data. If the data does not exist in cloud server, this uploader is the first uploader and uploads the data to the cloud server. Otherwise, this uploader is subsequent uploader and goes into the deduplication phase. The proof of ownership protocol aims to verify that the subsequent uploader indeed owns the original data. When the subsequent uploader passes the proof of ownership protocol, the cloud server distributes the symmetric key to subsequent uploader by using proxy re-encryption technology. Because the first uploader and subsequent uploader are data owners, they can decrypt the ciphertext to get the plaintext data. Moreover, the data owners authorize a third party auditor to audit the integrity of the cloud storage data.

5 Proposed Construction

In this section, we put forward a secure and efficient client-side data deduplication scheme with public auditing. The proposed scheme consists of five phases: initialization, upload, deduplication, decryption, and auditing.

5.1 The Initialization Phase

The system runs the Setup algorithm and generates the public parameters. Moreover, the CSP and the uploader generate their key pairs by running the KeyGen algorithm. In addition, uploaders interact with the key server to generate secret value of the data by running the secret value generation (SecValGen) algorithm. The details are as follows.

Setup: Let p, q be two large primes. Due to the property of safe primes, there exist two primes p' and q' that satisfy that $p = 2p' + 1$, $q = 2q' + 1$. We compute $n = p * q$ and choose generator g with order $\lambda = 2p'q'$, which can be chosen by selecting a random number $\varsigma \in Z^*_{n^2}$ and computing $g = -\varsigma^{2n}$. The value λ can be used for decryption, but we choose to conceal and protect it from all parties. In addition, the system chooses two groups G_1 and G_T of a prime order with bilinear map $e : G_1 \times G_1 \to G_T$. The system parameters are random generators $v \in G_1$ and $Z = e(v, v) \in G_T$. Then, it randomly chooses secure hash functions $H_1 : \{0,1\}^* \to Z^*_n$, $H_2 : \{0,1\}^* \to G_1$, $H_3 : Z^*_n \to \{0,1\}^*$. The system public parameters are $pars = (G_1, G_T, n, g, Z, H_1, H_2, H_3)$.

KeyGen: The CSP and the uploader j generates their key pairs: $(sk_{CSP}, pk_{CSP}) = (a, v^a)$ and $(sk_j, pk_j) = (u_j, v^{u_j})$ respectively. Besides, the uploader j selects a random number $x_{u_j} \in Z^*_n$ as his secret value.

SecValGen: The uploader interacts with the key server to generate a secret value of the data by the O-PRF protocol [21]. The O-PRF protocol is based on RSA blind signatures. The key server has secret key d and public key e where $ed \equiv 1 \bmod \Phi(n)$. $H_1 : \{0,1\}^* \to Z^*_n$ and $H_3 : Z^*_n \to \{0,1\}^*$ are two secure hash functions. The interaction process is as follows.

1) The uploader calculates the hash value $h = H_1(m)$ of the data and selects a random value $r \in Z^*_n$. Moreover, the uploader computes the blinded hash $x = h \cdot r^e \bmod n$ and sends x to the key server.

2) Upon receiving x, the key server computes $y = x^d \bmod n$ and sends y to the uploader.

3) The uploader calculates $y \cdot r^{-1} \bmod n$ and obtains secret value z. Finally, the uploader computes a secret value $\pi \leftarrow H_3(z)$.

Then, the uploader announces that it has a certain data via a tag. If the data does not exist in CSP, the uploader

goes into the upload phase. Otherwise, the uploader goes into the deduplication phase.

5.2 The Upload Phase

The first uploader performs an upload task that includes Encrypt algorithm and TagBlock algorithm. In this process, the first uploader generates the data ciphertext, the data tag, the ciphertext of symmetric key, the original proofs of data, and the auditing tag of data block. Finally, the first uploader uploads these information to the cloud server.

Encrypt: The encryption algorithm is divided into four parts, as shown below.

1) The first uploader generates the data ciphertext $E = Enc_k(m)$ using a random symmetric key k.

2) The first uploader computes the data tag $T = H_1(m)$.

3) The first uploader chooses two random values r_1 and r_2, and then encrypts symmetric key k using the public keys pk_{CSP} and the secret value π. The ciphertext of symmetric key k is denoted as: $C = \{C_1, C_2, C_3\} = \{(1 + k * n)g^{H_1(Z^{r_1})*r_2} \bmod n^2, g^{r_2} \bmod n^2, pk_{CSP}^{\pi r_1}\}$.

4) The first uploader divides the data m into n blocks (i.e., $m = \{m_1, \ldots, m_n\}$), and calculates the original proofs: $IPs_i = g^{-m_i} (\mathrm{mod} p)$, $IPs = (IPs_1, \ldots, IPs_n)$.

TagBlock: The first uploader computes data integrity tags. In particular, he splits E into n blocks (i.e., $E = \{E_1, E_2, \ldots, E_n\}$) and further divides each block E_i into s sectors (i.e., $E_i = \{E_{i,1}, E_{i,2}, \ldots, E_{i,s}\}$). Among them $g_1, g_2, \ldots, g_s$ are s elements in G_1. For each block E_i, the first uploader computes $T_i = [H_2(ID_C \parallel i) \cdot \prod_{j=1}^{s} g_j{}^{E_{i,j}}]^{\pi} \in G_1$.

5.3 The Deduplication Phase

If a data announced by the first uploader in the initialization phase exists in the cloud server, the subsequent uploaders go into the deduplication phase and run the proof of ownership protocol. When the subsequent uploader passes the proof of ownership protocol, the cloud server generates the re-encryption key and re-encrypts the ciphertext of symmetric key.

5.3.1 The Proof of Ownership Protocol

The proof of ownership protocol aims to provide a framework for the cloud server to verify that the subsequent uploader indeed owns the data rather than part of it. This phase includes the original proofs generation (OPsGen) algorithm, the coefficients generation (CosGen) algorithm, the final proofs generation (FPsGen) algorithm, and the proofs verification (ProVer) algorithm.

OPsGen: As shown in the encryption process, the first uploader generates the original proofs of the data: $IPs_i = g^{-m_i} (\mathrm{mod} p)$, $IPs = (IPs_1, \ldots, IPs_n)$. Then he sends these proofs to the cloud server. After receiving the original proofs, the cloud server aggregates these proofs using the aggregated proofs structure. As shown in Subsection 3.4.

CosGen: The subsequent uploaders utilize this algorithm to generate the coefficients according to the Schnorr's Identification Protocol [28]. Given μ random number r_i, $1 \le r_i \le q-1$, the subsequent uploaders compute the Coefficients: $commit_i = g^{r_i} (\mathrm{mod} p)$, $commit = (commit_1, \ldots, commit_\mu)$, and send these Coefficients to the cloud server.

FPsGen: The subsequent uploaders run this algorithm to generate the final proofs for cloud server's challenge. Let $Chal = (\theta, \gamma, a_1, a_2, \ldots, a_n)$, $1 \le \gamma \le 2^\alpha$ be selected uniformly at random, μ be the commit generated by Coefficients and $\theta, a_1, a_2, \ldots, a_n$ are a set of random integers. The cloud server allows the subsequent uploader to challenge the jth-level aggregated proofs and sends the challenge block index function $\varphi_\theta(\cdot)$ to the subsequent uploader. According to the block index function and index number, the subsequent uploader calculates the challenge set of the data. Then, the subsequent uploader calculates the final proofs:

$$\begin{aligned} FPs_t &= \gamma \sum_{i=1}^{2^{j-1}} a_i m_i + r_t (\mathrm{mod} q), \\ FPs &= (FPs_1, \ldots, FPs_{\lfloor \frac{n}{2^{j-1}} \rfloor}). \end{aligned}$$

ProVer: The cloud server receives the final proofs FPs from the subsequent uploader. According to the aggregated proofs structure, challenge set $Chal$ and Coefficients $commit$, the cloud server computes the product representation of the IPs and FPs: $DPs_t = g^{FPs_t} \times IPs_t{}^{\gamma} (\mathrm{mod} p)$. If $DPs_t = commit_t$, output true, the proof is recognized by the cloud server. Otherwise, output false, the proof is fake.

5.3.2 The Key Distribution Process

The cloud server generates the re-encryption key and the re-encryption ciphertext. This process includes the re-encryption key generation (RekGen) algorithm and the re-encryption (ReEnc) algorithm.

RekGen: The cloud server wants to delegate the subsequent uploader j by publishing re-encryption key $rk_{CSP \to j} = v^{u_j / a}$.

ReEnc: The cloud server computes ciphertext $C_3' = e(pk_{CSP}^{\pi r_1}, rk_{CSP \to j}) = Z^{\pi r_1 * u_j}$, and sets $C_2' = C_2$ and $C_1' = C_1$. Finally, the cloud server generates the re-encryption ciphertext $C' = \{C_1', C_2', C_3'\}$.

5.4 The Decryption Phase

When the data owner proposes a decryption request, the cloud server sends the ciphertext to the data owner. Then, the Decrypt algorithm is as follows.

Decrypt: Upon receiving the encrypted data tuple (E, C'), the data owner can directly decrypt it to obtain the original data. The specific steps for decryption are as follows.

1) The data owner computes $C_3'' = H_1((C_3')^{1/\pi u_j}) = H_1(Z^{r_1})$.

2) The data owner obtains the symmetric key $k = L(C_1/(C_2')^{C_3''} \bmod n^2)$ where $L(u) = (u-1)/n$.

3) The data owner obtains the data $m = Dec_k(E)$ using the symmetric key k.

5.5 The Auditing Phase

During the auditing process, the proposed scheme adopts the proxy re-signature technology to achieve efficient auditing. Firstly, the user computes re-signature keys. Secondly, the cloud service provider generates corresponding auditing proofs by using cloud storage data and re-signature keys. Finally, the third party auditor verifies the integrity of the target data by the user's public key and auditing proofs. This process includes the re-signature keys generation (Rekey) algorithm, the generation proof (GenProof) algorithm, and the check proof (CheckProof) algorithm.

Rekey: It is performed by user to compute re-signature keys, which enables the cloud to prove the integrity of the challenged data under user-associated private/public key pair. The user j computers $d_{u,m} = u_j \cdot (\pi)^{-1} + \beta$, $h_{u,m} = \beta \cdot x_{u_j}$ and also sets $rk_{u,m} = (d_{u,m}, h_{u,m})$, where β is a random number in Z_n^*.

GenProof: The third party auditor chooses a random c-element subset $I \subset [1, n]$ along with c random coefficients in Z_n^*. Let $Q = \{i, v_i\}_{i \in I}$ be the set of challenge index-coefficient pairs. After receiving Q from the third party auditor, the cloud server sends a proof $P = (\sigma, \sigma_1, \rho_1, \ldots, \rho_s)$ back to the third party auditor, where $\sigma = \prod_{(i,v_i)\in Q} T_i^{d_{u,m}\cdot v_i} \in G_1$, $\sigma_1 = \prod_{(i,v_i)\in Q} T_i^{h_{u,m}\cdot v_i} \in G_1$ and $\rho_j = \sum_{(i,v_i)\in Q} v_i \cdot E_{i,j} \in Z_n^*$ for $1 \le j \le s$.

CheckProof: The third party auditor sends σ_1 to user and obtains $\sigma_1' = \sigma_1^{x_u^{-1}}$. The third party auditor then accepts the proof if the following equation holds:

$$e(\frac{\sigma}{\sigma_1'}, g) = e(\prod_{(i,v_i)\in Q} H_2(ID_C \parallel i)^{v_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j}, pk_u)$$

6 Security Analysis

In this section, we analyze security of the proposed scheme. The security consists of two parts: The data deduplication phase and the data auditing phase.

6.1 Data Deduplication Phase

In this case, we mainly concentrate on the security of the PoW protocol, the O-PRF protocol, and the ciphertext of symmetric key.

Theorem 1. *The proposed proofs of ownership (PoW) protocol is a zero-knowledge proof of knowledge assuming that the discrete logarithm is hard.*

Proof. A zero-knowledge proof protocol satisfies the following three properties: completeness, soundness and zero-knowledge. Assuming that the discrete logarithm problem is hard means that no adversary can compute the secret value m_i from the original proofs IPs_i, where $IPs_i = g^{-m_i} (\mathrm{mod} p)$. In the process of aggregating proofs, the cloud server aggregates the original proofs into the j-level proofs. For the j-level proofs, no adversary can compute the secret information m_i. □

Completeness. Completeness means that a client has the original data blocks, and both the client and cloud server follow the instructions, then the cloud server must accept the client. This is because

$$\begin{aligned} DPs_t &= g^{FPs_t} \times (IPs_t)^{\gamma} \bmod p \\ &= g^{\gamma \sum_{i=1}^{2^{j-1}} a_i m_i + r_t} \cdot (g^{-\sum_{i=1}^{2^{j-1}} a_i m_i})^{\gamma} \bmod p \\ &= g^{r_t} \bmod p \\ &= commit_t \end{aligned}$$

Soundness. Soundness means that if a client does not have the original data blocks, then regardless of what the client does, the cloud server will pass the proofs with probability that it can be ignored. Assuming the client is a cheater, he does not have the correct original data blocks m_i. $commit_t$ is transmitted in iteration, the server, after picking $\gamma \in \{0,1\}^{\alpha}$, is waiting for:

$$FPs_t = \log_g(commit_t IPs_t{}^{\gamma} \bmod p) (\mathrm{mod} q)$$

This equation shows that, for fixed $commit_t$ and IPs_t, there will be 2^{α} distinct values for FPs_t which correspond to 2^{α} distinct values for e. So the client guesses probability for each $-\sum_{i=1}^{2^{j-1}} a_i m_i$ is $2^{-\alpha}$. Here let $\lfloor \frac{n}{2^{j-1}} \rfloor$ be equal to η. In PoW protocol, the client interacts with the server η times. If all the η commitments are admitted, the cloud server marks this client as the owner of this data. The false positive probability for the verify protocol is $2^{-\eta\alpha}$.

Zero-knowledge. For a perfect zero-knowledge proof protocol, which does not need to negotiate between

the prover and the verifier. We introduce a simulator that produces the proof transcript of simulation. During the proof of ownership of this document, the simulator effectively generates the proof transcript without interacting with the real client, and the transcript generated by these simulators is indistinguishable from the actual transcript.

For common input IPs_t, we can construct a polynomial-time (in $|p|$) simulator S as follows.

1) S initializes transcript as an empty string;
2) (a) S picks $FPs_t \in Z_q$; (b) S picks $\gamma \in \{0,1\}^\alpha$; FPs_t must be uniform in Z_q for either cases of $\gamma \in \{0,1\}^\alpha$ and independent of the common input IPs_t; (c) S computes $commit_t \leftarrow g^{FPs_t} IPs_t{}^\gamma \bmod p$; $commit$ must also be uniform and independent of the common input IPs_t; (d) $Transcript \leftarrow Transcript \parallel commit_t, \gamma, FPs_t$.

Clearly, $Transcript(commit_t, \gamma, FPs_t)$ can be produced by S in polynomial time, and the elements in it have distributions which are the same as those in a real proof transcript. Therefore, the protocol is perfect zero-knowledge.

In summary, the data sent from the client in a run is uniform, they can tell the cloud server that there is no information about the client's private input b_i. Regardless of how the server selects the random challenge bits, the elements in the client's records are uniform, so even if the cloud server is dishonest, the protocol is a perfect zero knowledge.

Theorem 2. *The O-PRF protocol is an interactive protocol between the uploader and the key server. The key server will not obtain the secret value π. In addition, during the O-PRF protocol, no information will be revealed.*

Proof. In the O-PRF protocol, the uploader blinds the hash value $H_1(m)$ of the data and sends it to the key server which can not obtain the hash value $H_1(m)$ of data. Therefore, the key server will not obtain the secret value π. In the process of interaction between the uploader and the key server, the uploader blinds the hash value $H_1(m)$ and sends x to the key server. The key server signs the blinded value x and sends y to the uploader. Because x and y are blinded by the random value r, the O-PRF protocol will not leak any information. □

Theorem 3. *If the DL problem holds in group G_1 and the CDH problem holds in group $Z^*_{n^2}$, then the ciphertext of symmetric key is secure in the proposed scheme.*

Proof. The ciphertext of the symmetric key is $C = \{C_1, C_2, C_3\} = \{(1 + k * n) g^{H_1(Z^{r_1}) * r_2} \bmod n^2, g^{r_2} \bmod n^2, pk_{CSP}^{\pi r_1}\}$. The cloud server and unauthorized users would like to obtain the symmetric key k. □

The secret value π is obtained by the O-PRF protocol between the uploader and the key server. According to Theorem 2, the cloud server can not obtain the secret value π. Because the DL problem is difficult, it is hard to get v^{r_1} from $pk_{CSP}^{\pi r_1} = v^{\pi r_1 a}$. Thus, the cloud server can not obtain the value of $H(Z^{r_1})$. The cloud server re-encrypts the ciphertext of the symmetric key, the obtained ciphertext is: $C' = \{C'_1, C'_2, C'_3\} = \{(1 + k * n) g^{H_1(Z^{r_1}) * r_2} \bmod n^2, g^{r_2} \bmod n^2, Z^{\pi r_1 * u_j}\}$. Unauthorized users with a secret value of π also can not obtain the value $H(Z^{r_1})$, because he can not obtain the private key u_j of user j. Bounded by the difficulty of the CDH problem, the cloud server and unauthorized users can not get $g^{H_1(Z^{r_1}) * r_2}$ from $g^{H_1(Z^{r_1})}$ and g^{r_2}. Hence, they can not obtain the symmetric key k. In addition, a malicious user who does not have a secret key d, therefore, cannot generate secret value π. It allows encryption to be secure against the brute force attacks even for predictable message set. Therefore, the key distribution of the proposed scheme is secure.

6.2 Data Auditing Phase

In this case, we focus on the correctness and unforgeability of the integrity auditing scheme.

Theorem 4. *The cloud server is able to generate a proof that passes the verification if all the challenged blocks and their integrity tags are correctly stored.*

Proof. Proving the correctness of our integrity auditing scheme for data is equivalent to proving that equation $e(\frac{\sigma}{\sigma'_1}, g) = e(\prod_{(i,v_i)\in Q} H_2(ID_C \parallel i)^{v_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j}, pk_u)$ hold. According to the properties of the bilinear map, the correctness can be verified by the following calculations.

$$
\begin{aligned}
& e(\tfrac{\sigma}{\sigma'_1}, g) \\
&= e(\textstyle\prod_{(i,v_i)\in Q} T_i^{d_{u,m} v_i} \cdot (\prod_{(i,v_i)\in Q} T_i^{r_{u,m} v_i})^{-1}, g) \\
&= e(\textstyle\prod_{(i,v_i)\in Q} T_i^{u_j \pi^{-1} v_i}, g) \\
&= e(\textstyle\prod_{(i,v_i)\in Q} [H_2(ID_C \parallel i) \cdot \prod_{j=1}^{s} g_j^{E_{i,j}}]^{u_j \cdot v_i}, g) \\
&= e(\textstyle\prod_{(i,v_i)\in Q} [H_2(ID_C \parallel i)^{v_i} \cdot \prod_{j=1}^{s} g_j^{v_i E_{i,j}}], pk_u) \\
&= e(\textstyle\prod_{(i,v_i)\in Q} [H_2(ID_C \parallel i)^{v_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j}], pk_u)
\end{aligned}
$$

□

Theorem 5. *Under the CDH assumption, the integrity auditing scheme is secure against an adaptive chosen-message attack in the random oracle model.*

Proof. Assuming that the CDH assumption holds in G. If there is a polynomial time adversary $\mathcal{A}$, he has the advantage $Adv_{\mathcal{A}}$ to break our scheme. Then, we show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to solve the CDH problem. That is, given a CDH tuple (g, g^a, g_0), the adversary $\mathcal{B}$ is able to compute g_0^a with non-negligible probability. In the process of proof, the adversary $\mathcal{B}$ is the challenger for the adversary $\mathcal{A}$. The process of proof is as follows. □

Setup: The normal user-associated public keys are set to be $pk_{nu} = g^{a s_{nu}}$ for $nu \in [1, l]$, where s_{nu} are randomly chosen from Z_q^*. Moreover, the adversary

$\mathcal{B}$ sets $g_j = g_0^{y_j}$ for $1 \le j \le s$. Besides, $\mathcal{B}$ chooses x_{nu} randomly from Z_q^*. For malicious users, $\mathcal{B}$ selects random numbers x_{mu}, s_{mu} $(mu \in [1, l'])$ and computes the public keys $pk_{mu} = g^{s_{mu}}$. Finally, the system parameters, the normal user public keys $pk_{nu}(nu \in [1, l])$, the malicious public and private key pairs $(pk_{mu}, s_{mu}, x_{mu})(mu \in [1, l'])$ are given to the adversary $\mathcal{A}$.

Query 1: There are four types of queries that $\mathcal{A}$ can request: oracle SecValGen, oracle Rekey, oracle TagBlock and the hash function H_1.

1) Oracle **SecValGen:** if π'_ω has not been queried before, $\mathcal{B}$ returns a random number $x'_w \in Z_q^*$ to $\mathcal{A}$ and records it in list DataKey. Otherwise, $\mathcal{B}$ obtains x'_w from list DataKey and responds it to $\mathcal{A}$.

2) Oracle **Rekey:** for malicious user public key pk_{mu}, the adversary $\mathcal{B}$ returns $(x'_w)^{-1} \cdot s_{mu} + r_{mu,w'} \cdot x_{mu}$, where $r_{mu,w'}$ is a random in Z_q^*. For normal user, $\mathcal{B}$ returns two random numbers to $\mathcal{A}$.

3) Oracle **TagBlock:** if $ID_{E'_i} \parallel V_i$ has not been queried before, the adversary $\mathcal{B}$ chooses a random element from G_1 as the value of $H_1(ID_{E'_i} \parallel V_i)$ and then computes $T'_i = [H_1(ID_{E'_i} \parallel V_i) \cdot \prod_{j=1}^{s} g_j^{E_{i,j}}]^{x'_w}$ for the query. Finally, $\mathcal{B}$ records T'_i in list and returns $H_1(ID_{E'_i} \parallel V_i)$ for the corresponding hash query. Otherwise, the adversary $\mathcal{B}$ returns T'_i from list to the adversary $\mathcal{A}$.

Query 2: There are three types of queries that $\mathcal{A}$ can request: oracle Rekey, oracle TagBlock and the hash function H_1.

1) Oracle **Rekey:** The adversary $\mathcal{B}$ returns $(x_w^{-1} \cdot s_{nu} + r_{nu,w}, x_{nu} r_{nu,w})$ to $\mathcal{A}$, where $r_{nu,w}$ is a random number in Z_q^*.

2) Oracle **TagBlock:** if $ID_{E'_i} \parallel V_i$ has not been queried before, the adversary $\mathcal{B}$ computes $T_i = g^{a x_w r_i}$ and records it in list. Finally, $\mathcal{B}$ returns T_i and $H_1(ID_{E'_i} \parallel V_i) = \frac{g^{r_i}}{\prod_{j=1}^{s} g_j^{E_{i,j}}}$ for the corresponding hash query. It is easily observed that T_i is a valid tag under the public key pk_π. If $\{E_i, ID_{E_i} \parallel V_i\}$ is in list, the adversary $\mathcal{B}$ obtains T_i and returns it to the adversary $\mathcal{A}$.

Challenge: The adversary $\mathcal{B}$ requests the adversary $\mathcal{A}$ to prove the integrity of all blocks $E_1, \ldots, E_n$ by sending coefficients $a_1, \ldots, a_n$ under the public key pk_u.

Forge: We assume that the adversary $\mathcal{A}$ has deleted or modified one or more blocks. Let $\rho'_j = \sum_{i=1}^{n} a_i E_{i,j}$ be the real result. The adversary $\mathcal{A}$ returns a proof $P = (\sigma, \sigma_1, \rho_1, \ldots, \rho_s)$ satisfying $e(\frac{\sigma}{\sigma'_1}, g) = e(\prod_{(i,v_i) \in Q} H_2(ID_C \parallel i)^{v_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j}, pk_u)$ but there exists at least one value $\rho_j = \rho'_j$. Since P is a valid proof under public key pk_u, we have

$$\begin{aligned}\frac{\sigma}{\sigma'_1} &= [\prod_{i=1}^{n} H_1(ID_C \parallel V_i)^{a_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j}]^{a s_u} \\ &= (\prod_{i=1}^{n} (H_1(\frac{g^{r_i}}{\prod_{j=1}^{s} g_j^{m_{i,j}}})^{a_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j})^{a s_u} \\ &= (\prod_{i=1}^{n} g^{r_i a_i} \cdot \prod_{j=1}^{s} g_j^{\rho_j - \rho'_j})^{a s_u} \\ &= g^{a s_u \sum_{i=1}^{n} r_i a_i} (g_0^{s_u \sum_{j=1}^{s} y_j (\rho_j - \rho'_j)})^a\end{aligned}$$

From the above equation, the adversary $\mathcal{B}$ can easily compute $g_0^a = (\frac{\sigma}{\sigma'_1 g^{a s_u \sum_{i=1}^{n} r_i a_i}})^{[s_u \sum_{j=1}^{s} y_j (\rho_j - \rho'_j)]^{-1}}$.

If the adversary $\mathcal{A}$ does not possess all the sectors $E_{i,j}$ $(1 \le i \le n, 1 \le j \le s)$, we analyze the probability that the adversary $\mathcal{A}$ successfully forges the values satisfying $P_j = P'_j$ for $(1 \le j \le s)$. Due to Theorem 2 in [20], we can know that the adversary $\mathcal{A}$ forges a valid value $P_j = P'_j$ is negligible.

Finally, if there is a polynomial time adversary $\mathcal{A}$ that has the advantage $Adv_{\mathcal{A}}$ to break our scheme, the adversary $\mathcal{B}$ can use $\mathcal{A}$ to solve the CDH problem. Since the CDH problem is a difficult problem, the probability that the adversary $\mathcal{A}$ breaks our scheme is negligible. Therefore, the proposed scheme is secure against an adaptive chosen-message attack in the random oracle model under the CDH assumption.

7 Performance Evaluation

In this section, we will conduct the performance evaluation including four aspects, the detection rate analysis, functionality comparison, efficiency comparison, and experimental comparison.

7.1 Detection Rate Analysis

Since Ding *et al.*'s scheme [6] and Liu *et al.*'s scheme [20] do not verify the data ownership based on the original data, we only make the detection rate analysis between Yang *et al.*'s scheme [28] and our scheme.

Suppose a client claims the ownership of an n-block data m, but actually he owns f out of n blocks of data m. Let's examine the probability that the cloud server accepts the client as the data owner. We use p_x to indicate the probability that the cloud server detects at least one missing block. We set x as the number of missing data blocks. During the proof of ownership process, if the missing data block on the client is not detected, the cloud server will accept the client's ownership of the data. Therefore, the probability that the client is accepted by the cloud server is $1 - p_x$. Since

$$p_x = p\{x \ge 1\} = 1 - \frac{C_{n-x}^{\mu}}{C_n^{\mu}} = 1 - \prod_{i=0}^{\mu-1} \frac{n-x-i}{n-i}$$

Based on the knowledge of probability theory, we can calculate: $1 - p_x \approx (1 - \frac{x}{n})^{\mu}$.

From the above equation, we can derive: $\mu \approx \left\lceil \log_{(1-\frac{x}{n})}(1 - p_x) \right\rceil$.

Table 1: Functionality comparison between our scheme and other schemes

Schemes	Ding *et al.* [6]	Liu *et al.* [20]	Yang *et al.* [28]	Our scheme
Secure proof of ownership	No	No	Yes	Yes
High detection rate	No	No	No	Yes
Data owner offline	Yes	No	No	Yes
No authorized party	No	No	Yes	Yes
Against brute-force attacks	Yes	No	Yes	Yes
Public auditing	No	Yes	No	Yes
One tag for each block	No	Yes	No	Yes

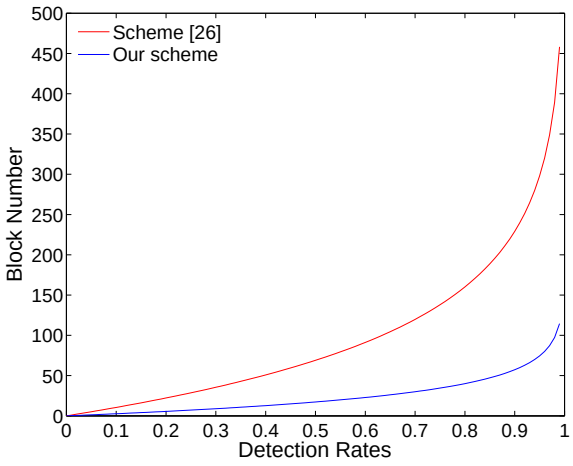

Figure 5: Challenged block numbers vary with detection rates

The proposed scheme aggregates original proofs and verifies the subsequent proofs. With the same number of verifications, we compare the detection rate of our scheme and Yang *et al.*'s scheme [28]. Figure 5 is the relationship between the detection rate and the number of challenges in our scheme and Yang *et al.*'s scheme [28] in the case of missing blocks rates of 1%. In the proposed scheme, we utilize the third-level aggregate proofs to verify ownership of the data. It can be seen that the detection rate of the proposed scheme is higher than Yang *et al.*'s scheme [28] with the same of data blocks.

7.2 Functionality Comparison

The functionality comparisons between the proposed scheme and the related schemes [6, 20, 28] are showed in Table 1.

Table 1 shows that the proposed scheme supports secure proof of ownership, high detection rate, data owner offline, no authorized party, against brute-force attacks and public auditing, while others only support partial functionality. By adopting the zero-knowledge proof technology and the aggregated proofs structure, the proposed scheme achieves the secure proof of ownership and high detection rate for the missing data block, respectively. Meanwhile, we combine the oblivious pseudo-random function protocol with proxy re-encryption technology to implement key distribution without online data owners or the authorized party. In addition, because the O-PRF protocol is used in the encryption process, the proposed scheme is secure against brute-force attacks. The proposed scheme utilizes a third party auditor for performing public auditing. Moreover, our scheme only generates one tag for each data block.

7.3 Efficiency Comparison

In this subsection, we will conduct efficiency comparisons including three aspects, the proof of ownership, the key distribution, and the public auditing. Since Liu *et al.*'s scheme [20] does not support client-side deduplication, we only make a comparison between the proposed scheme and the related schemes [6, 28] in proof of ownership and key distribution processes, respectively. Moreover, we conduct a comparison between the proposed scheme and Liu *et al.*'s scheme [20] in public auditing process, because Yang *et al.*'s scheme [28] and Ding *et al.*'s scheme [6] do not support public auditing.

As shown in the table below. $Pair$: bilinear pairing; Exp: exponentiation in G_1 or G_T; $ModExp$: modular exponentiation; $ModMul$: modular multiplication; n: the number of blocks; μ: the number of challenging blocks by the cloud server; j: the level of aggregated proofs; c: the number of subsequent uploaders; s: the number of sectors for each block; d: the number of challenging blocks by the third party auditor.

7.3.1 The Proof of Ownership Process

We make an efficiency comparison between the proposed scheme and the related schemes [6, 28] in Table 2. In ownership verification process, the proposed scheme and Yang *et al.*'s scheme [28] use zero knowledge proof technology, and Ding *et al.*'s scheme [6] utilizes bilinear pairing operation.

As shown in Table 2, the computation consumption of Ding *et al.*'s scheme [6] is much smaller than Yang *et al.*'s scheme [28] and the proposed scheme. However, Ding *et al.*'s scheme [6] verifies the data ownership based on the hash value. In other words, a malicious user could pass the PoW verification of client side deduplication if he could

Table 2: Efficiency comparison in proof of ownership

Schemes	Ding *et al.* [6]	Yang *et al.* [28]	Our scheme
OPs computation	Exp	$nModExp$	$nModExp$
FPs computation	Exp	$\mu ModMul$	$\frac{\mu}{j}ModMul$
Verification of μ blocks	$Pair$	$2\mu Exp+\mu ModMul$	$\frac{2\mu}{j}Exp+\frac{\mu}{j}ModMul$
Total computational costs	$2Exp+Pair$	$2\mu Exp+nModExp+2\mu ModMul$	$\frac{2\mu}{j}Exp+nModExp+\frac{2\mu}{j}ModMul$

get the hash value of the data. In addition, because of adopting the aggregated proofs structure, the computation consumption of the proposed scheme is smaller than Yang *et al.*'s scheme [28].

7.3.2 The Key Distribution Process

We make an efficiency comparison between the proposed scheme and the related schemes [6, 28] in Table 3 with regard to first uploader, CSP, subsequent uploader and AP. For the three comparison schemes, they use a symmetric encryption algorithm to encrypt the data, and then distribute the symmetric keys through the O-PRF protocol and proxy re-encryption technology. When comparing the efficiency of the three schemes, we ignore the symmetric encryption.

As shown in Table 3, Yang *et al.*'s scheme [28] incurs higher computation overhead than the proposed scheme and Ding *et al.*'s scheme [6] in the total computational costs. Meanwhile, the first uploader has a large computational overhead in Yang *et al.*'s scheme [28]. In addition, we also compare our scheme with Ding *et al.*'s scheme [6]. Although the computational cost of our scheme is slightly higher than Ding *et al.*'s scheme [6], our scheme does not require the introduction of an authorized party to complete key distribution. In the proposed scheme, the cloud server takes on the main computational costs and other entities have a little computational costs. As we all know, the cloud server's computing power can be considered infinitely, so the proposed scheme is more practical and effective.

7.3.3 The Public Auditing Process

Because Ding *et al.*'s scheme [6] and Yang *et al.*'s scheme [28] do not support public auditing, we only make an efficiency comparison between the proposed scheme and Liu *et al.*'s scheme [20]. In public auditing process, the proposed scheme and Liu *et al.*'s scheme [20] adopt the proxy re-signature technology to verify integrity of the cloud storage data. Moreover, these two schemes only generate one auditing tag for each data block.

As shown in Table 4, the computation consumption of the proposed scheme is higher than Liu *et al.*'s scheme [20]. However, because the proposed scheme utilizes the O-PRF protocol to generate secret values for calculating auditing tag, the proposed scheme can achieve better security.

7.4 Experimental Comparison

By utilizing the Pairing Based Cryptography (PBC) Library, an efficiency experiment result is given under the Linux environment. The following experiments run on a personal computer with its configuration parameters as Intel Core i5 2.5 GHz Processor and 4 GB RAM. The number of subsequent uploaders range from 10 to 50. The experiment includes five aspects, the computation cost of the first uploader, the CSP, the subsequent uploader, the AP, and the total computation cost. The experiment result given below comes from the average of 50 experiments.

As shown in Figure 6, we first evaluate the computation cost of the first uploader in key distribution process. With the same number of subsequent uploaders, the time cost of Ding *et al.*'s scheme [6] and our scheme is much less than Yang *et al.*'s scheme [28]. Figure 7 indicates that Ding *et al.*'s scheme [6], Yang *et al.*'s scheme [28] and our scheme have almost the same time overhead. In addition, we can see that the time cost of Yang *et al.*'s scheme [28] is much more than Ding *et al.*'s scheme [6] and our scheme in Figure 8. Since Yang *et al.*'s scheme [28] and our scheme do not introduce an authorized party to complete key distribution, we only show the time cost of Ding *et al.*'s scheme [6] in Figure 9. As shown in Figure 10, we compare the total computation cost in key distribution process. The computation time of Yang *et al.*'s scheme [28] is far more than Ding *et al.*'s scheme [6] and our scheme. Moreover, during the key distribution process, Ding *et al.*'s scheme [6] and Yang *et al.*'s scheme [28] require the assistance of online data owners and the authorized party, respectively. Therefore, our scheme is secure and efficient in the key distribution process.

8 Conclusions

In this paper, we have proposed a secure and efficient client side deduplication scheme with public auditing. We utilize zero-knowledge proof and aggregates proofs structure to achieve high detection rate of client missing blocks. Meanwhile, the proposed scheme achieves key distribution by the O-PRF protocol and proxy re-encryption technology. In addition, all data owners of the proposed scheme can audit cloud storage data by employing proxy re-signing technology. The security analysis shows that the proof of ownership scheme is sound, complete and zero-knowledge.

Table 3: Efficiency comparison in key distribution

Entities	Algorithm	Ding *et al.* [6]	Yang *et al.* [28]	Our scheme
First uploader	Setup	$1Exp$	$2Exp$+$1Pair$	$1Exp$
	Data upload	$2Exp$+$2ModExp$	$2cExp$	$2Exp$+$4ModExp$
	Rekey generation	–	$cExp$	–
CSP	System setup	–	–	$1Exp$
	Re-encryption	$cPair$	$cPair$	$cPair$
	Rekey generation	–	–	$cExp$
Subsequent uploaders	System setup	$cExp$	$2cExp$+$cPair$	$cExp$
	Decrypt ciphertext	$cExp$+$cModExp$	$cExp$	$cExp$+$3cModExp$
AP	System setup	$1Exp$	–	–
	Rekey generation	$cExp$	–	–
Total computational costs		$(c+2)ModExp$+ $(3c+4)Exp$+$cPair$	$(6c+2)Exp$+ $(2c+1)Pair$	$(3c+4)ModExp$+ $(3c+4)Exp$+$cPair$

Table 4: Efficiency comparison in public auditing

Schemes	Liu *et al.* [20]	Our scheme
Tag computation	$n(s+1)Exp$	$n(s+1)Exp$+$2ModExp$
Proof computation	$2dExp$	$2dExp$
Check proof	$d(s+1)Exp$+$2Pair$	$d(s+1)Exp$+$2Pair$
Total computational costs	$(ns+n+ds+3d)Exp$+$2Pair$	$(ns+n+ds+3d)Exp$+$2Pair$+$2ModExp$

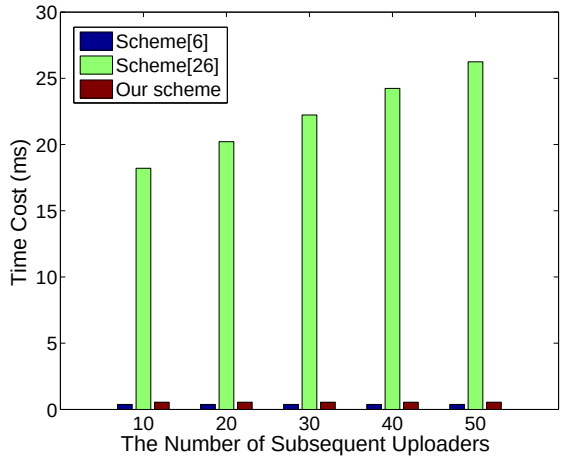

Figure 6: The computation cost of the first uploader

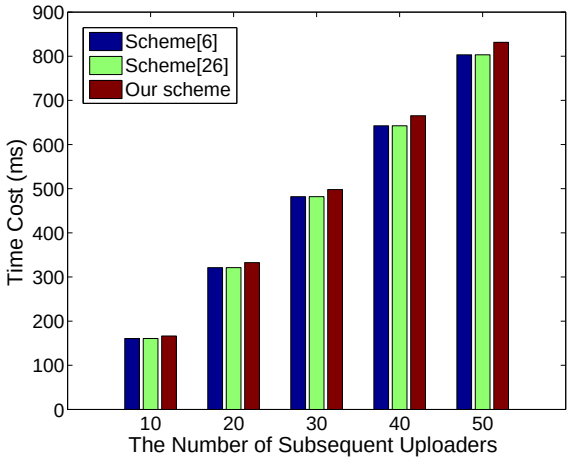

Figure 7: The computation cost of the CSP

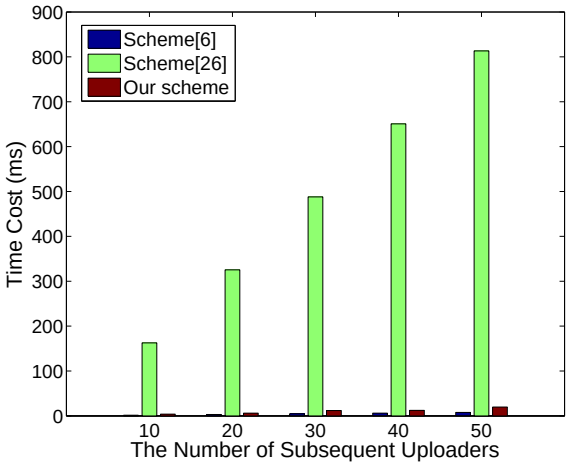

Figure 8: The computation cost of the subsequent uploaders

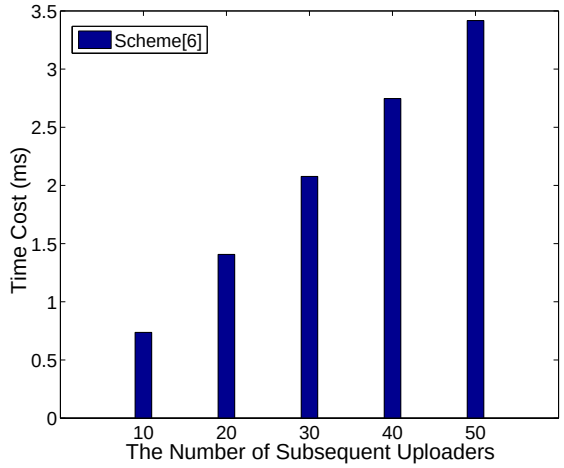

Figure 9: The computation cost of the AP

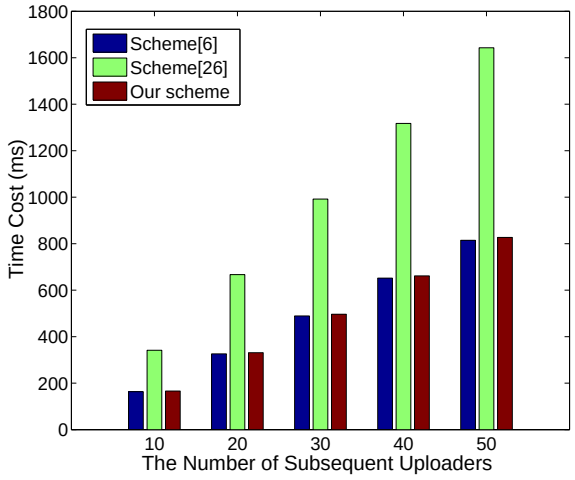

Figure 10: The total computation cost

Our scheme can protect the clients' symmetric key from being recovered by the server and other collusive clients for key distribution. In addition, our auditing scheme demonstrates the correctness and unforgeability. Finally, performance evaluation shows that the proposed scheme is practical and efficient.

Acknowledgments

We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China under Grants No.61472470 and 61702401.

References

[1] A. Agarwala, P. Singh, and P. K. Atrey, "Dice: A dual integrity convergent encryption protocol for client side secure data deduplication," in *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2176–2181, Oct. 2017.

[2] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: server-aided encryption for deduplicated storage," in *Usenix Conference on Security*, pp. 179–194, Aug. 2013.

[3] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 7881, pp. 296–312, 2013.

[4] Z. Cao, L. Liu, and O. Markowitch, "Analysis of one scheme for enabling cloud storage auditing with verifiable outsourcing of key updates," *International Journal of Network Security*, vol. 19, no. 6, pp. 950–954, 2017.

[5] S. Deshpande and R. Ingle, "Evidence based trust estimation model for cloud computing services," *International Journal of Network Security*, vol. 20, no. 2, pp. 291–303, 2018.

[6] W. Ding, Z. Yan, and R. H. Deng, "Secure encrypted data deduplication with ownership proof and user revocation," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 297–312, Aug. 2017.

[7] C. C. Erway, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security (TISSEC'15)*, vol. 17, no. 4, pp. 1–29, 2015.

[8] L. Gonzalez-Manzano and A. Orfila, "An efficient confidentiality-preserving proof of ownership for deduplication," *Journal of Network and Computer Applications*, vol. 50, pp. 49–59, 2015.

[9] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *ACM Conference on Computer and Communications Security*, pp. 491–500, Oct. 2011.

[10] D. Harnik, B. Pinkas, and A. Shulmanpeleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security and Privacy*, vol. 8, no. 6, pp. 40–47, 2010.

[11] W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for secure data storage in cloud computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133–142, 2016.

[12] M. S. Hwang, C. C. Lee, and T. H. Sun, "Data error locations reported by public auditing in cloud storage service," *Automated Software Engineering*, vol. 21, no. 3, pp. 373–390, 2014.

[13] M. S. Hwang, T. H. Sun, and C. C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *Journal of Circuits, Systems and Computers*, vol. 26, no. 5, 2017.

[14] K. Kim, T. Y. Youn, N. S. Jho, and K. Y. Chang, "Client-side deduplication to enhance security and reduce communication costs," *Etri Journal*, vol. 39, no. 1, pp. 116–123, 2017.

[15] L. Lei, Q. Cai, B. Chen, and J. Lin, "Towards efficient re-encryption for secure client-side deduplication in public clouds," in *International Conference on Information and Communications Security*, pp. 71–84, Nov. 2016.

[16] C. Li, H. Cheung, and C. Yang, "Secure and efficient authentication protocol for power system computer networks," *International Journal of Network Security*, vol. 20, no. 2, pp. 337–344, 2018.

[17] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2386–2396, 2016.

[18] C. W. Liu, W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing," *International Journal of Network Security*, vol. 18, no. 4, pp. 650–666, 2016.

[19] H. Liu, H. Ning, Y. Zhang, and L. T. Yang, "Aggregated-proofs based privacy-preserving authentication for v2g networks in the smart grid," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1722–1733, 2012.

[20] X. Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, "One-tag checker: Message-locked integrity auditing on encrypted cloud deduplication storage," in *IEEE Conference on Computer Communications*, pp. 1–9, May 2017.

[21] P. Mell and T. Grance, "Draft nist working definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, pp. 50–50, 2009.

[22] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage*, vol. 7, no. 4, pp. 1–20, 2012.

[23] S. Rezaei, M. Ali Doostari, and M. Bayat, "A lightweight and efficient data sharing scheme for cloud computing," *International Journal of Electronics and Information Engineering*, vol. 9, no. 2, pp. 115–131, 2018.

[24] Z. Wang, Y. Lu, G. Sun, "A policy-based deduplication mechanism for securing cloud storage," *International Journal of Electronics and Information Engineering*, vol. 2, no. 2, pp. 70–79, 2015.

[25] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, "Rse-pow: A role symmetric encryption pow scheme with authorized deduplication for multimedia data," *Mobile Networks and Applications*, vol. 23, no. 3, pp. 650–663, 2018.

[26] J. Xiong, Y. Zhang, L. Lin, J. Shen, X. Li, and M. Lin, "ms-PoSW: A multi-server aided proof of shared ownership scheme for secure deduplication in cloud," *Concurrency and Computation Practice and Experience*, no. 5, 2017.

[27] C. Yang, J. Ren, and J. Ma, "Provable ownership of files in deduplication cloud storage," *Security and Communication Networks*, vol. 8, no. 14, pp. 2457–2468, 2015.

[28] C. Yang, M. Zhang, Q. Jiang, J. Zhang, D. Li, J. Ma, and J. Ren, "Zero knowledge based client side deduplication for encrypted files of secure cloud storage in smart cities," *Pervasive and Mobile Computing*, vol. 41, pp. 243–258, 2017.

[29] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *Communications and Network Security*, pp. 145–153, Oct. 2013.

[30] J. Zhang, P. Li, and M. Xu, "On the security of an mutual verifiable provable data auditing in public cloud storage," *International Journal of Network Security*, vol. 19, no. 4, pp. 605–612, 2017.

Biography

Qianlong Dang is a master degree student in the School of Mathematics and Statistics at Xidian University. His interest focuses on cryptography and network security.

Hua Ma is a professor in the School of Mathematics and Statistics at Xidian University, Xi'an, China. Her research includes security theory and technology in electronic commerce design and analysis of fast public key cryptography theory and technology of network security.

Zhenhua Liu is a professor in the School of Mathematics and Statistics at Xidian University, Xi'an, China. His research interests include public key cryptography, cryptographic theory and security protocols in cloud computing.

Ying Xie is a master degree student in the School of Mathematics and Statistics at Xidian University. Her interest focuses on cryptography and network security.